Eurocrypt 2025

Committing Authenticated Encryption: Generic Transforms with Hash Functions

Shan Chen[1]

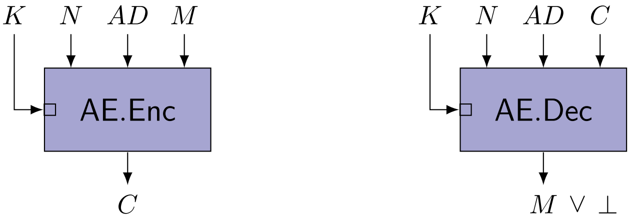
Vukašin Karadžić[2]

[1] Southern University of Science and Technology, Shenzhen, China

[2] Technische Universität Darmstadt, Germany

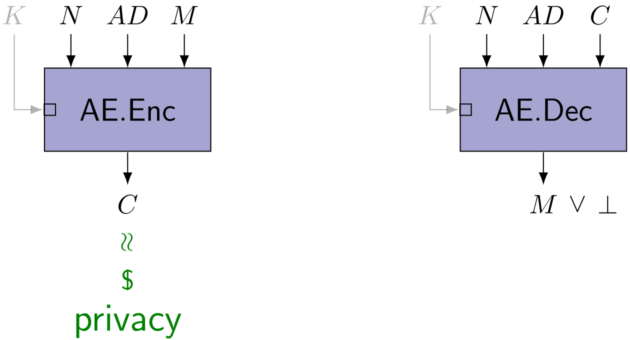
Authenticated Encryption and Committing Security

Authenticated Encryption



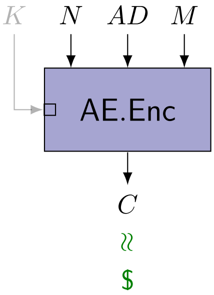
K	key
N	nonce
AD	associated data
M	message
C	ciphertext

Authenticated Encryption

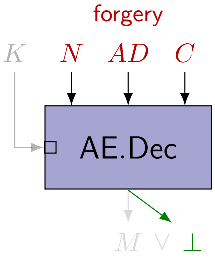


K	key
N	nonce
AD	associated data
M	message
C	ciphertext

Authenticated Encryption



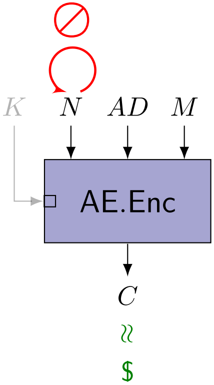
privacy



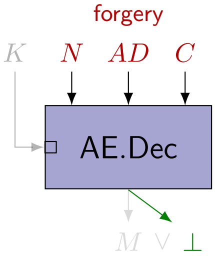
authenticity

K	key
N	nonce
AD	associated data
M	message
C	ciphertext

Authenticated Encryption



privacy

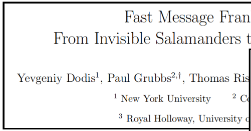


authenticity

K	key
N	nonce
AD	associated data
M	message
C	ciphertext

Authenticated Encryption and Committing Security

Authenticated Encryption and Committing Security



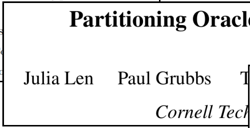
Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris

[1] New York University [2] C

[3] Royal Holloway, University c

[DGRW18]



Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tecl

[LGR21]

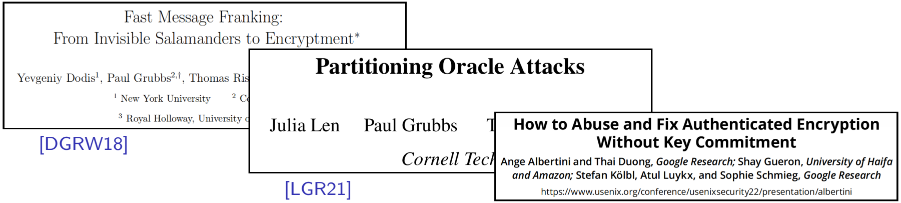
How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

[ADG$^+$22]

Authenticated Encryption and Committing Security



Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris

[1] New York University [2] C

[3] Royal Holloway, University

[DGRW18]

Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tec

[LGR21]

How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

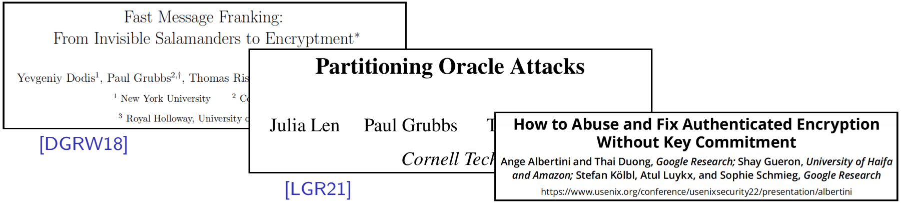
https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

[ADG$^+$22]

CMT(-3/4) [CR22, BH22]

$$(K, N, AD, M) \neq (K', N', AD', M')$$

Authenticated Encryption and Committing Security



Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris
[1] New York University [2] C
[3] Royal Holloway, University o

[DGRW18]

Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tech

[LGR21]

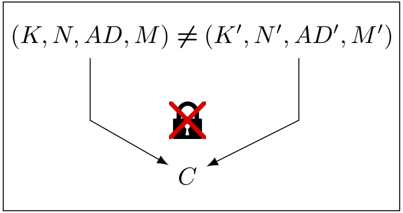
How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

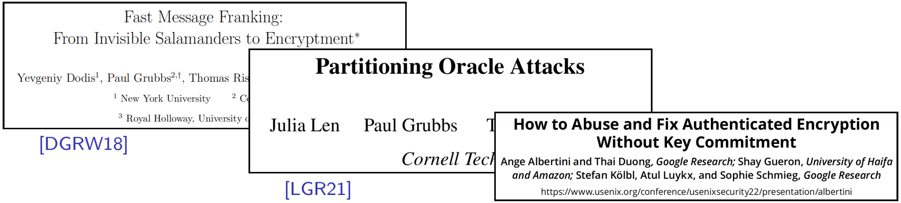
[ADG$^+$22]

CMT(-3/4) [CR22, BH22]



$$(K, N, AD, M) \neq (K', N', AD', M')$$

Authenticated Encryption and Committing Security



Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris

[1] New York University [2] C

[3] Royal Holloway, University o

[DGRW18]

Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tech

[LGR21]

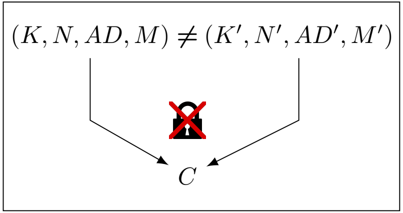
How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

[ADG+22]

CMT(-3/4) [CR22, BH22]

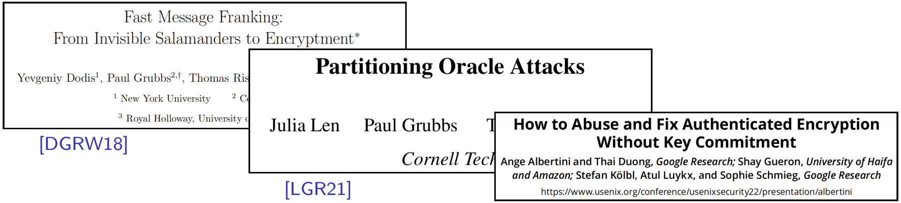


$$(K, N, AD, M) \neq (K', N', AD', M')$$

Previous work:

- popular and deployed AE schemes not committing: AES-GCM, OCB, ChaCha20/Poly1305, etc. [GLR17, DGRW18, LGR21, ADG+22]

Authenticated Encryption and Committing Security



Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris

[1] New York University [2] C

[3] Royal Holloway, University o

[DGRW18]

Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tec

[LGR21]

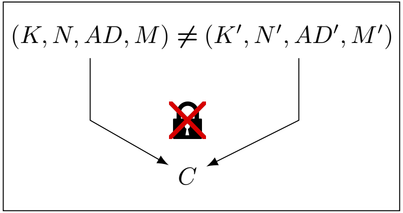
How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

[ADG+22]

CMT(-3/4) [CR22, BH22]

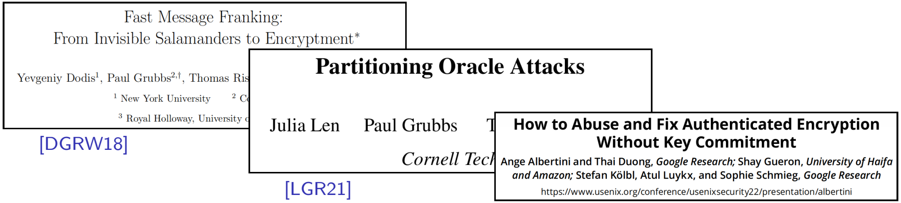


$$(K, N, AD, M) \neq (K', N', AD', M')$$

Previous work:

- popular and deployed AE schemes not committing: AES-GCM, OCB, ChaCha20/Poly1305, etc. [GLR17, DGRW18, LGR21, ADG+22]
- dedicated modifications (e.g., [BH22])

Authenticated Encryption and Committing Security



Fast Message Franking:
From Invisible Salamanders to Encryptment*

Yevgeniy Dodis[1], Paul Grubbs[2,†], Thomas Ris

[1] New York University [2] C

[3] Royal Holloway, University o

[DGRW18]

Partitioning Oracle Attacks

Julia Len Paul Grubbs T

Cornell Tech

[LGR21]

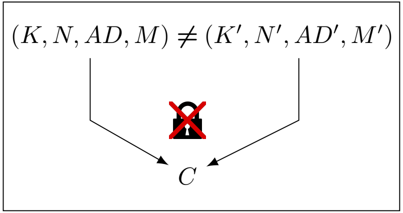
How to Abuse and Fix Authenticated Encryption Without Key Commitment

Ange Albertini and Thai Duong, *Google Research;* Shay Gueron, *University of Haifa and Amazon;* Stefan Kölbl, Atul Luykx, and Sophie Schmieg, *Google Research*

https://www.usenix.org/conference/usenixsecurity22/presentation/albertini

[ADG$^+$22]

CMT(-3/4) [CR22, BH22]



$$(K, N, AD, M) \neq (K', N', AD', M')$$

Previous work:

- popular and deployed AE schemes not committing: AES-GCM, OCB, ChaCha20/Poly1305, etc. [GLR17, DGRW18, LGR21, ADG$^+$22]
- dedicated modifications (e.g., [BH22])
- **generic transforms** (e.g., [ADG$^+$22, BH22, BCC$^+$24])

Our Motivation

- Existing generic transforms have one of the following shortcomings:

Our Motivation

- Existing generic transforms have one of the following shortcomings:
  1. not committing to the entire encryption context (i.e., the whole (K, N, AD, M) tuple)

For example:

CommitKeyΠ [ADG$^+$22]

only key-committing

Context commitment naming stems from [MLGR23].

Our Motivation

- Existing generic transforms have one of the following shortcomings:
  1. not committing to the entire encryption context (i.e., the whole (K, N, AD, M) tuple)
  2. involving non-standard primitives

For example:

SIV [BCC+24]

key-committing MAC

Context commitment naming stems from [MLGR23].

Our Motivation

- Existing generic transforms have one of the following shortcomings:
  1. not committing to the entire encryption context (i.e., the whole (K, N, AD, M) tuple)
  2. involving non-standard primitives
  3. not a black-box transform

For example:

CTX [CR22]

“tag-based” AE

Context commitment naming stems from [MLGR23].

Our Motivation

- Existing generic transforms have one of the following shortcomings:
  1. not committing to the entire encryption context (i.e., the whole (K, N, AD, M) tuple)
  2. involving non-standard primitives
  3. not a black-box transform
  4. provide limited committing security

For example:

PACT/comPACT [BBD24]

committing tag
from blockcipher

Context commitment naming stems from [MLGR23].

Our Motivation

- Existing generic transforms have one of the following shortcomings:
  1. not committing to the entire encryption context (i.e., the whole (K, N, AD, M) tuple)
  2. involving non-standard primitives
  3. not a black-box transform
  4. provide limited committing security

$\Rightarrow$ Investigate how to achieve **committing AE** using **black-box** generic transforms with **standard primitives**

Context commitment naming stems from [MLGR23].

Choosing Building Blocks

(Authenticated) Encryption

Choosing Building Blocks

(**Authenticated**) **Encryption**

- look at both plain privacy-only encryption (E) and authenticated encryption (AE) schemes

  → *crypto-agility*

Choosing Building Blocks

(Authenticated) Encryption

- look at both plain privacy-only encryption (E) and authenticated encryption (AE) schemes

  → *crypto-agility*

Hash Functions

Choosing Building Blocks

(Authenticated) Encryption

- look at both plain privacy-only encryption (E) and authenticated encryption (AE) schemes

  → *crypto-agility*

Hash Functions

- to achieve CMT-secure AE: idealized assumption, like *ideal cipher* or *random oracle* model is currently unavoidable (for practical instantiations)

Choosing Building Blocks

(Authenticated) Encryption

- look at both plain privacy-only encryption (E) and authenticated encryption (AE) schemes

  → *crypto-agility*

Hash Functions

- to achieve CMT-secure AE: idealized assumption, like *ideal cipher* or *random oracle* model is currently unavoidable (for practical instantiations)
- we opt-out for hash functions (and random oracle model):

Choosing Building Blocks

(Authenticated) Encryption

- look at both plain privacy-only encryption (E) and authenticated encryption (AE) schemes

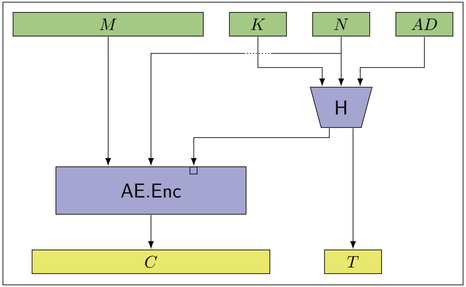
  → *crypto-agility*

Hash Functions

- to achieve CMT-secure AE: idealized assumption, like *ideal cipher* or *random oracle* model is currently unavoidable (for practical instantiations)
- we opt-out for hash functions (and random oracle model):
  - known and widely deployed primitive
  - easily gives us committing property (*collision resistance*)
  - CMT security can easily be increased by taking longer digest

Our Transforms: HtAE

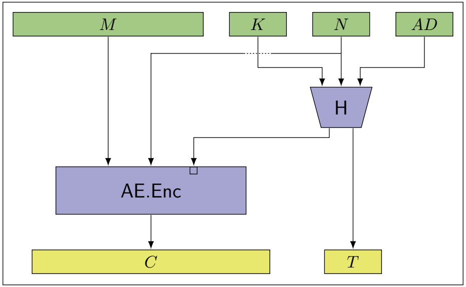
HtAE
(Hash-then-AE)



Our Transforms: HtAE

HtAE

(Hash-then-AE)

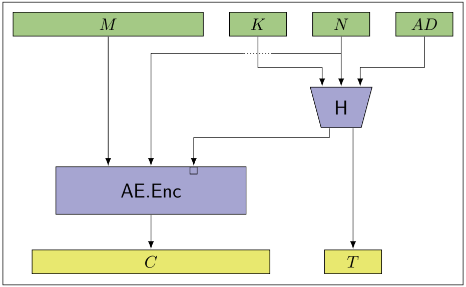


- HtAE **rekeys** underlying AE for every encryption query

Our Transforms: HtAE

HtAE

(Hash-then-AE)

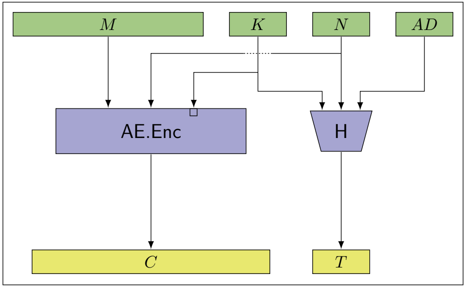


- HtAE **rekeys** underlying AE for every encryption query

costly, but still similar performance in comparison to existing transforms that rekey internally

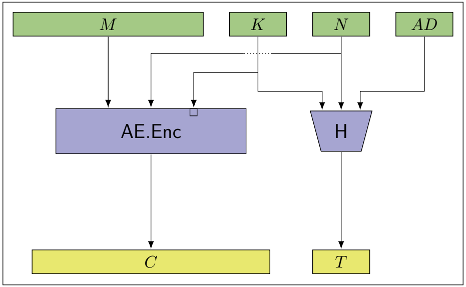
Our Transforms: AEaH

AEaH
(AE-and-Hash)



Our Transforms: AEaH

AEaH
(AE-and-Hash)

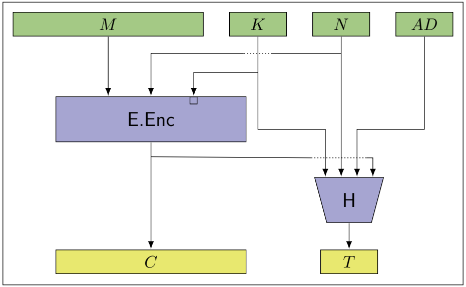


- AEaH is *fully* **parallelizable**

Our Transforms: EtH

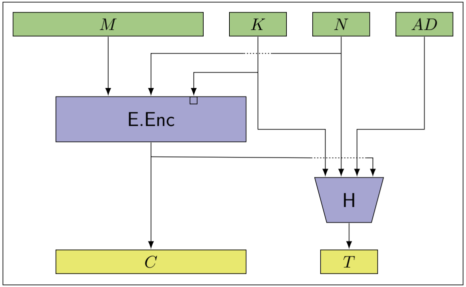
EtH

(Encrypt-then-Hash)



Our Transforms: EtH

EtH
(Encrypt-then-Hash)

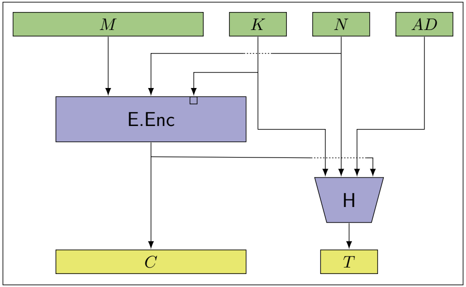


- Encryption primitive E only privacy-secure

Our Transforms: EtH

EtH

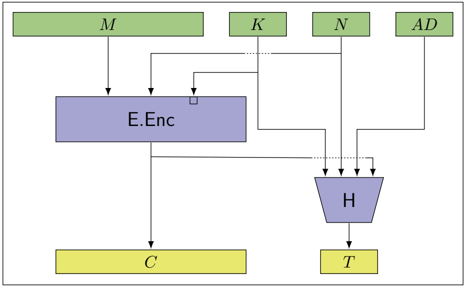
(Encrypt-then-Hash)



- Encryption primitive E only privacy-secure, but can also be AE

Our Transforms: EtH

EtH
(Encrypt-then-Hash)

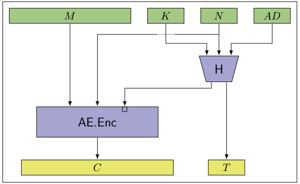


- Encryption primitive E only privacy-secure, but can also be AE

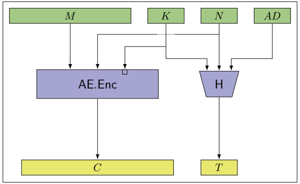
crypto-agility

Security Results

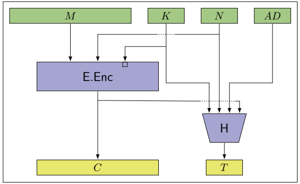
HtAE



AEaH



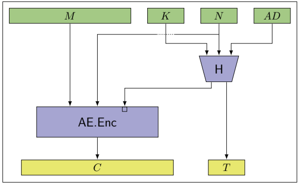
EtH



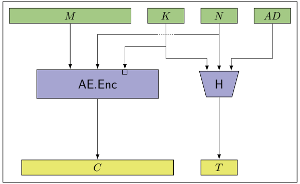
- Our HtAE, AEaH and EtH transforms are:

Security Results

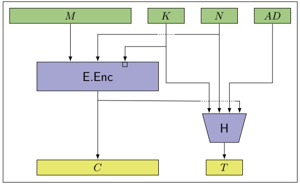
HtAE



AEaH



EtH

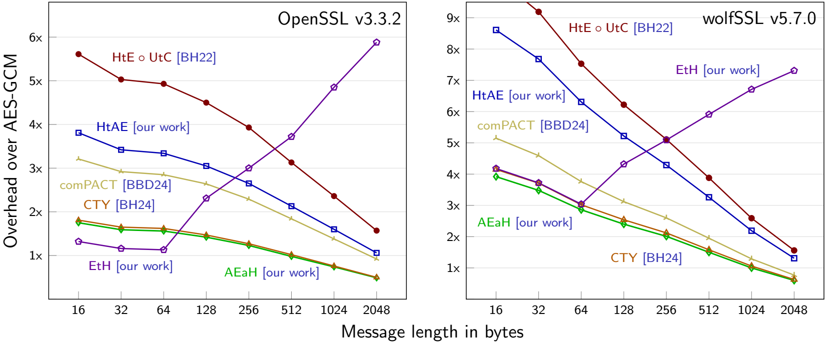


- Our HtAE, AEaH and EtH transforms are:

privacy, authenticity and CMT-secure

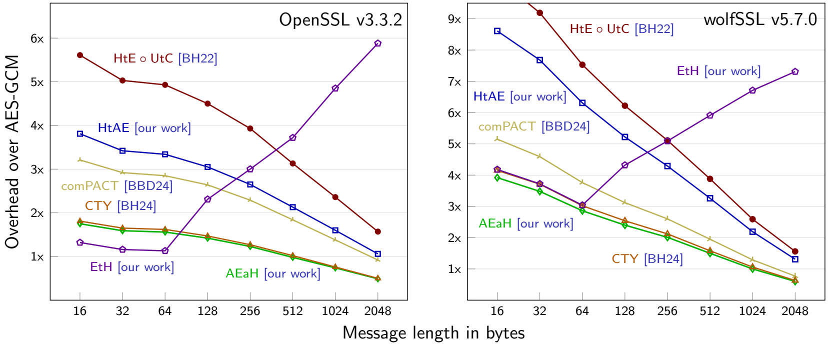
(Encryption) Performance Evaluation

(Encryption) Performance Evaluation



Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

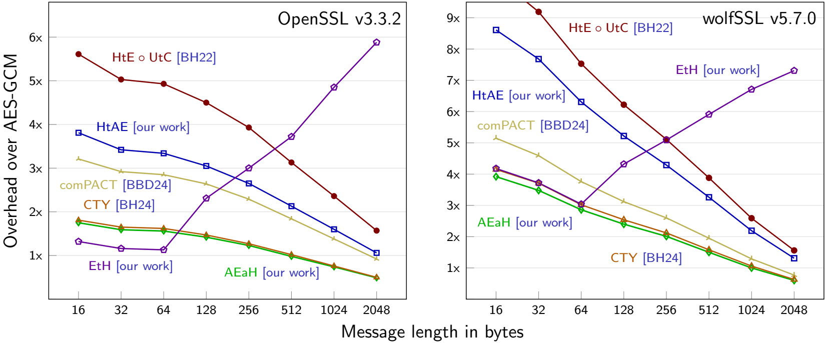
(Encryption) Performance Evaluation



Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

- recommendation: use AEaH if you aim for efficiency, and EtH if you have short messages and/or only access to E.

(Encryption) Performance Evaluation

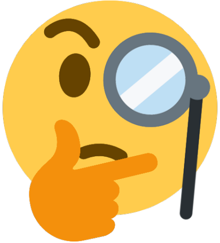


Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

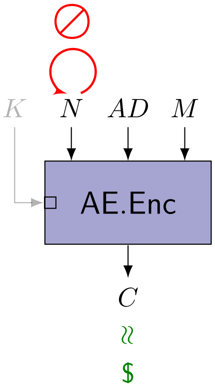
- recommendation: use AEaH if you aim for efficiency, and EtH if you have short messages and/or only access to E.
- here our (*parallelizable*) AEaH is implemented sequentially; dedicated implementation would perform even better

What if ...

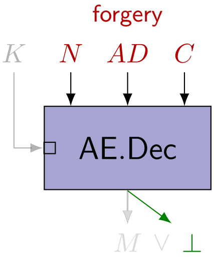
... a nonce repeats in the encryption?



What if a Nonce Repeats in the Encryption?



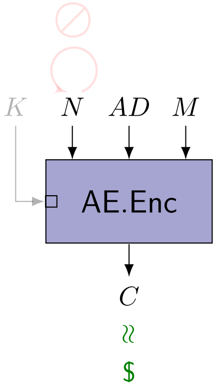
privacy



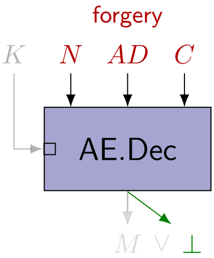
authenticity

K	key
N	nonce
AD	associated data
M	message
C	ciphertext

What if a Nonce Repeats in the Encryption?



privacy



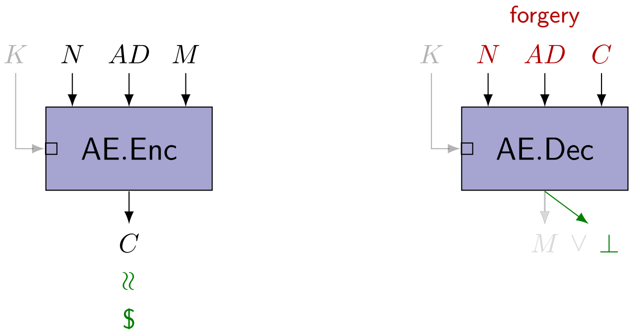
authenticity

K	key
N	nonce
AD	associated data
M	message
C	ciphertext

What if a Nonce Repeats in the Encryption?

(nonce) misuse-resistant authenticated encryption

MRAE



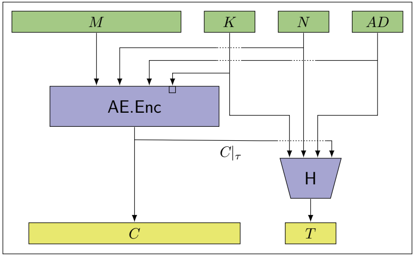
privacy

authenticity

K	key
N	nonce
AD	associated data
M	message
C	ciphertext

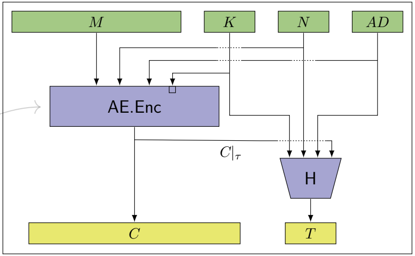
MRAE-Preserving Transform: AEtH

AEtH



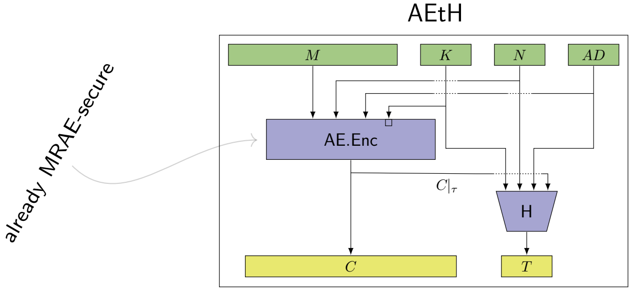
MRAE-Preserving Transform: AEtH

AEtH



already MRAE-secure

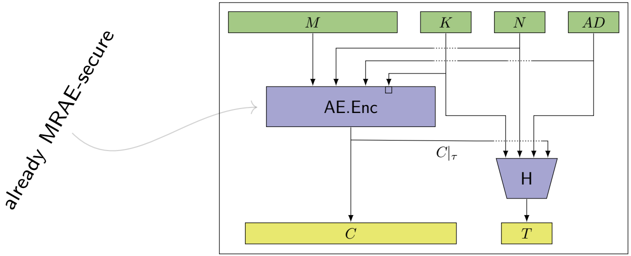
MRAE-Preserving Transform: AEtH



- Black-box generalization of CTX [CR22] (authors of [CR22] did not show MRAE security)

MRAE-Preserving Transform: AEtH

AEtH

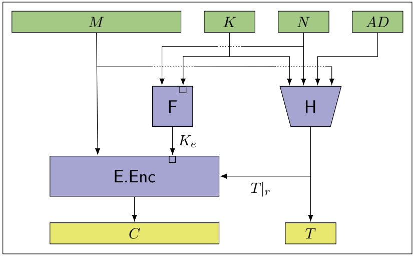


- Black-box generalization of CTX [CR22] (authors of [CR22] did not show MRAE security)

$$\boxed{\text{MRAE-secure AE} + \text{coll. res. H} \xRightarrow{\text{ROM}} \text{AEtH is MRAE- and CMT-secure}}$$

MRAE-Lifting Transform: chaSIV

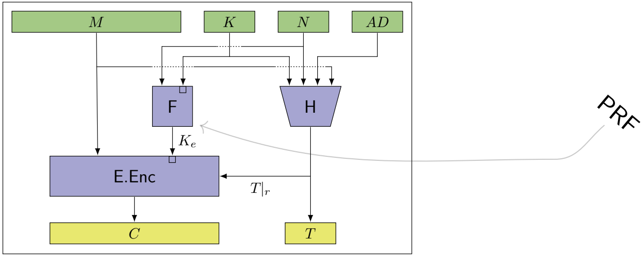
chaSIV



- **c**ommitting **ha**sh-based **SIV**

MRAE-Lifting Transform: chaSIV

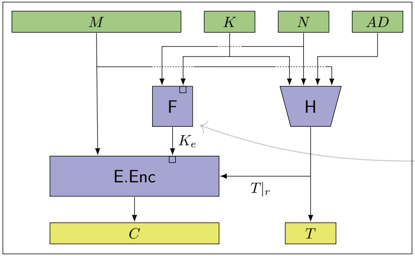
chaSIV



- **c**ommitting **ha**sh-based **SIV**

MRAE-Lifting Transform: chaSIV

chaSIV



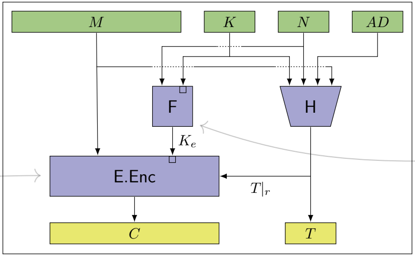
could be inst. with AES, so no new primitive implementation is needed

PRF

- **c**ommitting **ha**sh-based **SIV**

MRAE-Lifting Transform: chaSIV

chaSIV



only privacy-secure

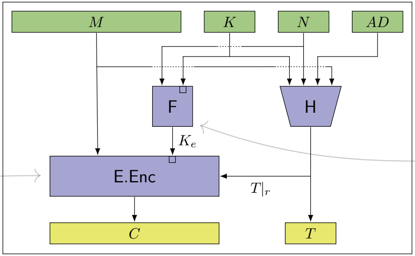
PRF

could be inst. with AES, so no new primitive implementation is needed

- **c**ommitting **ha**sh-based **SIV**

MRAE-Lifting Transform: chaSIV

chaSIV

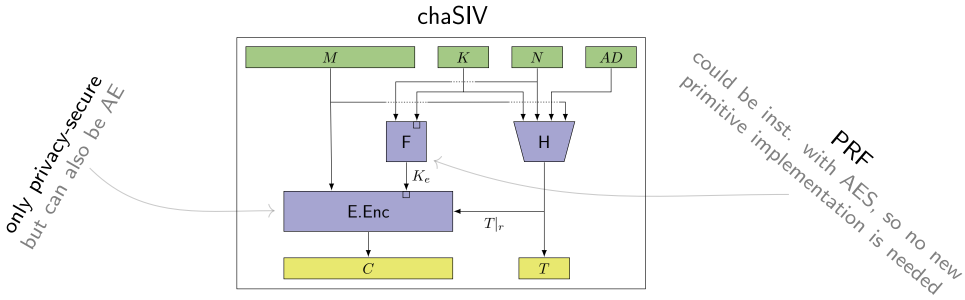


only privacy-secure
but can also be AE

PRF
could be inst. with AES, so no new
primitive implementation is needed

- **c**ommitting **ha**sh-based **SIV**

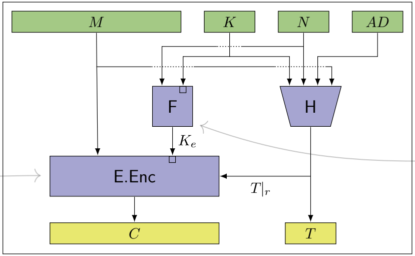
MRAE-Lifting Transform: chaSIV



- **c**ommitting **ha**sh-based **SIV**
- *first generic transform* that promotes plain E to **committing MRAE-secure** scheme

MRAE-Lifting Transform: chaSIV

chaSIV



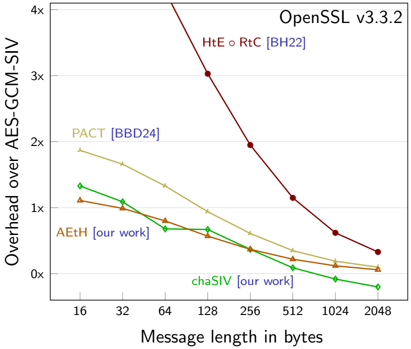
only privacy-secure
but can also be AE

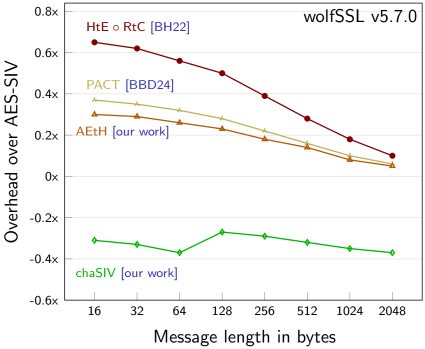
could be inst. with AES, so no new primitive implementation is needed
PRF

- **c**ommitting **ha**sh-based **SIV**
- *first generic transform* that promotes plain E to **committing MRAE-secure** scheme

privacy-secure $\mathsf{E} + $ coll. res. $\mathsf{H} + $ PRF $\mathsf{F} \xrightarrow{\text{ROM}}$ chaSIV is MRAE- and CMT-secure

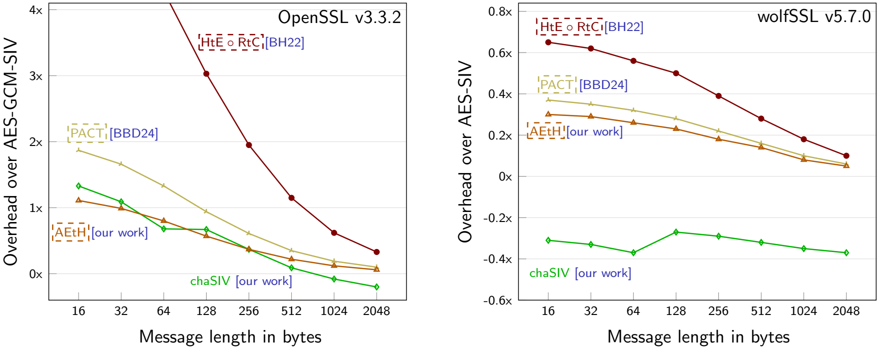
(Encryption) Performance Evaluation: MRAE-secure Transforms





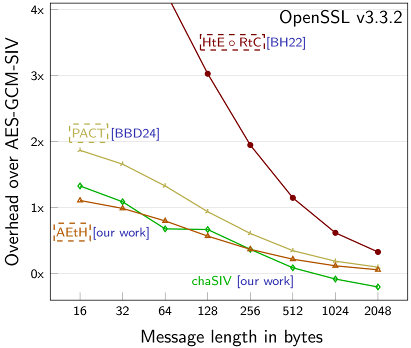
Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

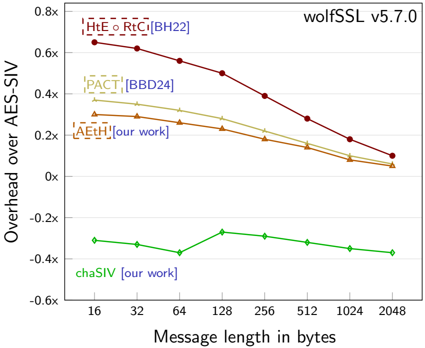
(Encryption) Performance Evaluation: MRAE-secure Transforms



Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

(Encryption) Performance Evaluation: MRAE-secure Transforms

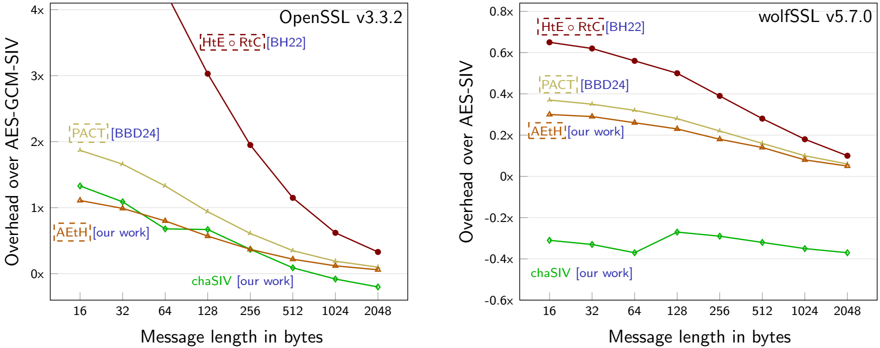




Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

- MRAE-preserving transforms: our AEtH (*black-box generalization of* CTX) performs the best

(Encryption) Performance Evaluation: MRAE-secure Transforms

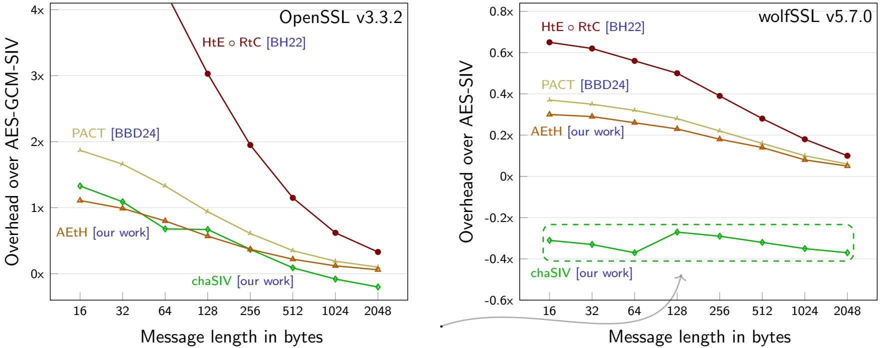


Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

- MRAE-preserving transforms: our AEtH (*black-box generalization of* CTX) performs the best

  additionally: CTX decryption algorithm would need *two* passes using OpenSSL's API, and would even be *impossible* to implement in wolfSSL

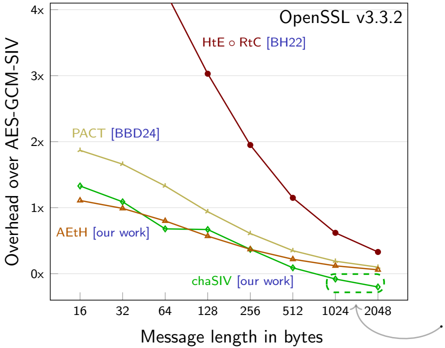
(Encryption) Performance Evaluation: MRAE-secure Transforms

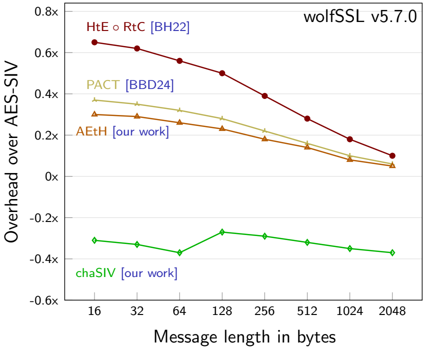


Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

- In wolfSSL: AEtH beats the benchmark AES-SIV for all message lengths

(Encryption) Performance Evaluation: MRAE-secure Transforms





Note: *all transforms are implemented using only* **black-box** *primitive implementations of the underlying library*

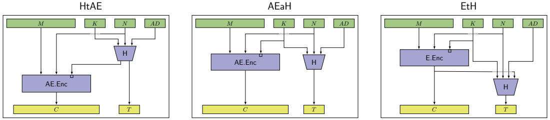
- In wolfSSL: AEtH beats the benchmark AES-SIV for all message lengths
- In OpenSSL: AEtH beats the benchmark AES-GCM-SIV for long messages

Takeaway

Takeaway

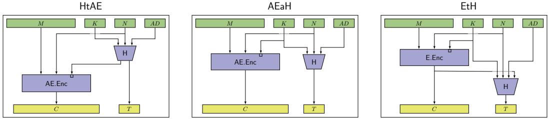
Takeaway

3 basic committing AE-secure transforms


HtAE
AEaH
EtH

Takeaway

3 basic committing AE-secure transforms

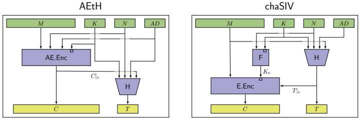


HtAE

AEaH

EtH

2 advanced committing MRAE-secure transforms

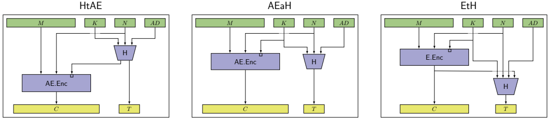


AEtH

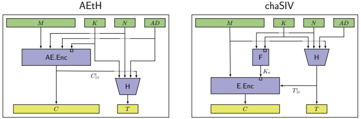
chaSIV

Takeaway

3 basic committing AE-secure transforms



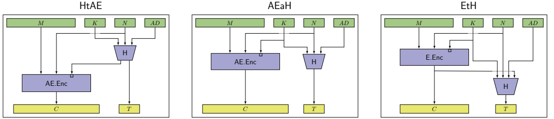
2 advanced committing MRAE-secure transforms



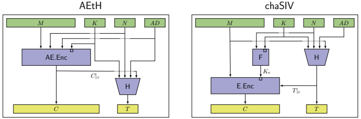
- easy to grasp and implement (*standardized primitives*)

Takeaway

3 basic committing AE-secure transforms



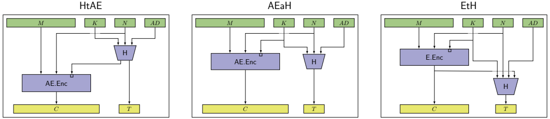
2 advanced committing MRAE-secure transforms



- easy to grasp and implement (*standardized primitives*)
- our transforms, implemented only with **black-box** primitives from common cryptographic libraries, are very **efficient**

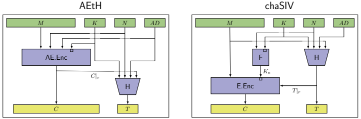
Takeaway

3 basic committing AE-secure transforms



HtAE AEaH EtH

2 advanced committing MRAE-secure transforms

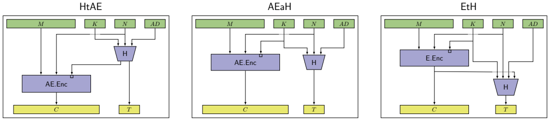


AEtH chaSIV

- easy to grasp and implement (*standardized primitives*)
- our transforms, implemented only with **black-box** primitives from common cryptographic libraries, are very **efficient**
- hope: **fast adoption of committing AEAD**
- please contact us if you're interested in our work

Takeaway

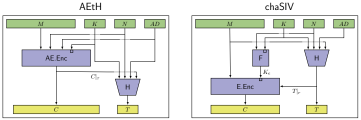
3 basic committing AE-secure transforms


HtAE

AEaH

EtH

2 advanced committing MRAE-secure transforms


AEtH

chaSIV

- easy to grasp and implement (*standardized primitives*)
- our transforms, implemented only with **black-box** primitives from common cryptographic libraries, are very **efficient**
- hope: **fast adoption of committing AEAD**
- please contact us if you're interested in our work

Thanks!

Questions?

many more details (e.g. CDY security) → ePrint

Code Artifact ← IACR Results Reproduced

References I

[ADG$^+$22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg.
How to abuse and fix authenticated encryption without key commitment.
In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 3291–3308, Boston, MA, USA, August 10–12, 2022. USENIX Association.

[BBD24] Arghya Bhattacharjee, Ritam Bhaumik, and Chandranan Dhar.
Universal context commitment without ciphertext expansion.
Cryptology ePrint Archive, Report 2024/1382, 2024.

[BCC$^+$24] Ritam Bhaumik, Bishwajit Chakraborty, Wonseok Choi, Avijit Dutta, Jérôme Govinden, and Yaobin Shen.
The committing security of MACs with applications to generic composition.
In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IV*, volume 14923 of *Lecture Notes in Computer Science*, pages 425–462, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[BH22] Mihir Bellare and Viet Tung Hoang.
Efficient schemes for committing authenticated encryption.
In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[BH24] Mihir Bellare and Viet Tung Hoang.
Succinctly-committing authenticated encryption.
In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IV*, volume 14923 of *Lecture Notes in Computer Science*, pages 305–339, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

References II

[CR22] John Chan and Phillip Rogaway.
On committing authenticated-encryption.
In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022: 27th European Symposium on Research in Computer Security, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 275–294, Copenhagen, Denmark, September 26–30, 2022. Springer, Cham, Switzerland.

[DGRW18] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage.
Fast message franking: From invisible salamanders to encryptment.
In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart.
Message franking via committing authenticated encryption.
In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

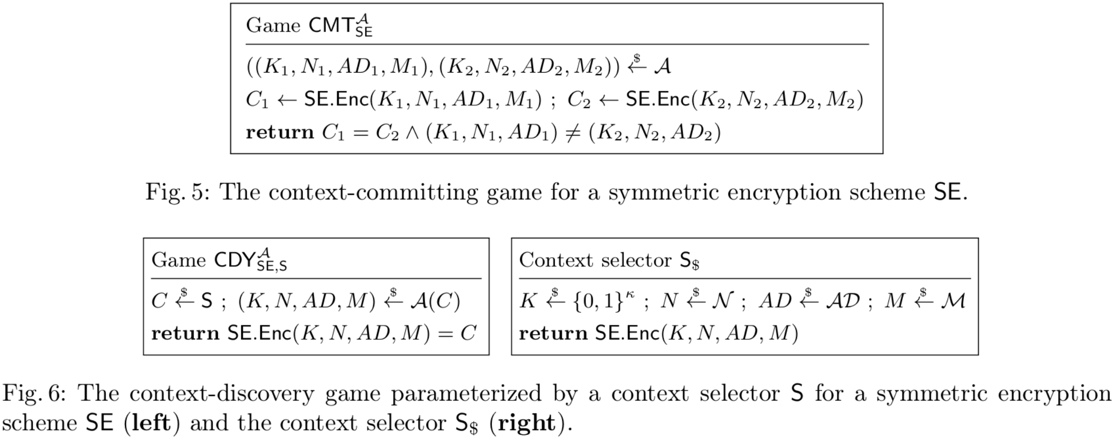
[LGR21] Julia Len, Paul Grubbs, and Thomas Ristenpart.
Partitioning oracle attacks.
In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 195–212. USENIX Association, August 11–13, 2021.

References III

[MLGR23] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart.
Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more.
In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 379–407, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

Backup Slides

CMT and CDY Security Notions



```
Game $\mathsf{CMT}^{\mathcal{A}}_{\mathsf{SE}}$
$((K_1, N_1, AD_1, M_1), (K_2, N_2, AD_2, M_2)) \xleftarrow{\$} \mathcal{A}$
$C_1 \leftarrow \mathsf{SE.Enc}(K_1, N_1, AD_1, M_1)$ ; $C_2 \leftarrow \mathsf{SE.Enc}(K_2, N_2, AD_2, M_2)$
return $C_1 = C_2 \wedge (K_1, N_1, AD_1) \neq (K_2, N_2, AD_2)$
```

Fig. 5: The context-committing game for a symmetric encryption scheme SE.

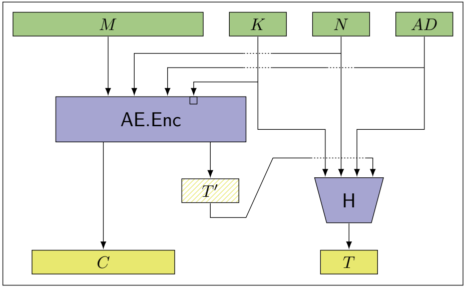
```
Game $\mathsf{CDY}^{\mathcal{A}}_{\mathsf{SE,S}}$
$C \xleftarrow{\$} \mathsf{S}$ ; $(K, N, AD, M) \xleftarrow{\$} \mathcal{A}(C)$
return $\mathsf{SE.Enc}(K, N, AD, M) = C$
```

```
Context selector $\mathsf{S}_{\$}$
$K \xleftarrow{\$} \{0,1\}^{\kappa}$ ; $N \xleftarrow{\$} \mathcal{N}$ ; $AD \xleftarrow{\$} \mathcal{AD}$ ; $M \xleftarrow{\$} \mathcal{M}$
return $\mathsf{SE.Enc}(K, N, AD, M)$
```

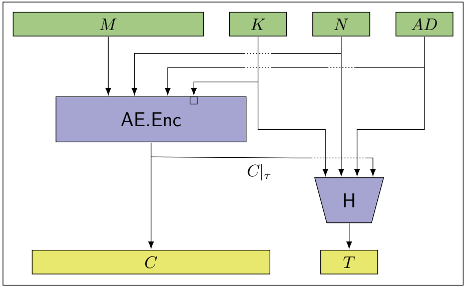
Fig. 6: The context-discovery game parameterized by a context selector S for a symmetric encryption scheme SE (**left**) and the context selector $\mathsf{S}_{\$}$ (**right**).

CTX vs. black-box generalization AEtH

CTX [CR22]



AEtH [our work]



Performance Evaluation Details: AE-secure Transforms

- Implemented *encryption* algorithm of our transforms and existing transforms

  $$\mathsf{HtE} \circ \mathsf{UtC}\ \text{[BH22]},\ \mathsf{CTY}\ \text{[BH24]},\ \mathsf{comPACT}\ \text{[BBD24]}$$

  that target *the same* (i.e., strongest) CMT security in OpenSSL and wolfSSL:

  - Use AES-GCM as AE in both libraries
  - Use AES-CTR as E in OpenSSL and AES-GCM as E in wolfSSL
  - Use truncated SHA-512 as H

- Performance measured as overhead over baseline (non-committing) AES-GCM speed.

  Testing setup: Intel Core i5-8265U CPU (Skylake microarchitecture), with the base frequency of 1.6GHz and the hyper-threading, frequency scaling and turbo mode functionalities disabled

Performance Evaluation Details: MRAE-secure Transforms

- Implemented *encryption* algorithm of our and existing MRAE-secure transforms

  $$\mathsf{HtE} \circ \mathsf{RtC}$$ [BH22], CTX [CR22], PACT [BBD24]

  that target *the same* (i.e., strongest) CMT security in OpenSSL and wolfSSL:

  - Use AES-GCM-SIV as AE in OpenSSL and AES-SIV in wolfSSL
  - Use AES-CTR as E in OpenSSL and AES-GCM as E in wolfSSL
  - Use truncated SHA-512 as H + plain AES as F

- Performance measured as overhead over baseline (non-committing) AES-GCM-SIV / AES-SIV speed.

  Testing setup: Intel Core i5-8265U CPU (Skylake microarchitecture), with the base frequency of 1.6GHz and the hyper-threading, frequency scaling and turbo mode functionalities disabled